IN THE CLAIMS

Please amend claims 1, 11, 23, 25 and 28, and cancel claims 2, 13 and 26, all without prejudice, as indicated on the following listing of all the claims in the present application after this Amendment.

1. (Currently amended) A computer processing method comprising the acts of:

using a shared pipeline instruction datapath and a shared pipeline processing unit to execute a first pipeline under a first context;

receiving a request to execute under a second context and, in response to the request, enabling a second pipeline to execute under the second context, wherein the enabling occurs during execution of the first pipeline, the enabling including ~~fetching an address vector associated with the second context~~ suspending execution of the first pipeline for a clock cycle and using the clock cycle to fetch an address vector associated with the second context;

detecting a halt in execution of the first pipeline; and

using the shared datapath and shared processing unit to execute the second pipeline after detecting the halt.

2. (Canceled)

3. (Original) The method of claim 1, wherein the halt is a first pipeline thread end.

4. (Original) The method of claim 1, wherein the halt is a direct memory access stall.

5. (Original) The method of claim 1, wherein the halt is due to the second context being assigned a higher priority than the first context.

6. (Original) The method of claim 1, wherein executing the second pipeline comprises flushing instructions associated with the first context from registers of the shared instruction datapath.

7. (Original) The method of claim 1, wherein executing the second pipeline comprises flushing instructions associated with the first context from registers of the shared processing unit.

8. (Original) The method of claim 1, wherein executing the second pipeline occurs only if a first pipeline status allows context switching.

9. (Original) The method of claim 1 further comprising the acts of:

detecting a halt in execution of the second pipeline; and

using the shared instruction datapath and the shared processing unit to resume execution of the first context subsequent to detecting the halt in execution of the second context.

10. (Original) The method of claim 1 further comprising the act of executing a debug context without disturbing execution of the first or second pipelines.

11. (Currently amended) A context switching microprocessor comprising:

a shared pipeline instruction datapath comprising a pipeline address stage, a pipeline fetch stage, a pipeline memory stage, and a pipeline decode stage;

a shared pipeline processing unit;

a first set of pipeline registers associated with a first pipeline;

a second set of pipeline registers associated with a second pipeline;

a control multiplexer coupling the first set of pipeline registers to the shared datapath and to the shared processing unit, and coupling the second set of pipeline registers to the shared datapath and to the shared processing unit; and

a set of shared registers associated with both the first pipeline and the second pipeline.

12. (Original) The microprocessor of claim 11 further comprising a pipeline controller coupled to the control multiplexer and to the shared instruction datapath.

13. (Canceled)

14. (Original) The microprocessor of claim 11, wherein the shared processing unit comprises a pipeline execution stage.

15. (Original) The microprocessor of claim 11 further comprising a memory controller coupled to the shared instruction data path and to the shared processing unit.

16. (Original) A context switching microprocessor comprising:

a shared instruction datapath;

a shared processing unit comprising a first set of processing registers associated with executing a first context pipeline, a second set of processing registers associated with executing a second context pipeline, and a set of shared registers associated with executing both the first and the second context pipelines, the shared processing unit being coupled to the shared instruction datapath; and

a context control data storage comprising a first context register associated with executing the first context pipeline and a second context register associated with executing the second context pipeline, the context control data storage being coupled to the shared instruction datapath and to the shared processing unit.

17. (Original) The microprocessor of claim 16, wherein the shared instruction datapath comprises a pipeline address stage, a pipeline fetch stage, a pipeline memory stage, and a pipeline decode stage.

18. (Original) The microprocessor of claim 16, wherein the first context register comprises a first program counter register associated with the first pipeline context and the second context register comprises a second program counter register associated with the second pipeline context.

19. (Original) The microprocessor of claim 16, wherein the shared registers comprise at least one register associated with cyclic redundancy checking.

20. (Original) The microprocessor of claim 16, wherein the context control data storage comprises a register associated with executing a debug context.

21. (Original) The microprocessor of claim 16 further comprising a pipeline controller coupled between the context control data storage and the shared instruction datapath, and coupled between the context control data storage and the shared processing unit.

22. (Original) The microprocessor of claim 21 further comprising a request controller coupled to the pipeline controller.

23. (Currently amended) A communication processing system comprising:

a communication engine comprising a pipeline context switching microprocessor, the context switching microprocessor comprising:

a shared instruction datapath;

a shared processing unit comprising a first set of processing registers associated with executing a first context pipeline, a second set of processing registers associated with executing a second context pipeline, and a set of shared registers associated with executing both the first and the second context pipelines, the shared processing unit being coupled to the shared instruction datapath; and

a context control data storage comprising a first context register associated with executing the first context pipeline and a second context register associated with executing the second context pipeline, the context control data storage being coupled to the shared instruction datapath and to the shared processing unit; and

a system block comprising a system microprocessor coupled to the communication engine.

24. (Original) The system of claim 23, wherein the system is formed as a single integrated circuit.

25. (Currently amended) The system of claim 23, wherein the context switching microprocessor further comprises:

~~a shared pipeline instruction datapath;~~

~~a shared pipeline processing unit;~~

~~a first set of pipeline registers;~~

~~a second set of pipeline registers; and~~

a control multiplexer coupling the first set of ~~pipeline~~ processing registers to the shared datapath ~~and to the shared processing unit~~, and coupling the second set of ~~pipeline~~ processing registers to the shared datapath and to the shared processing unit.

26. (Canceled)

27. (Previously presented) The computer processing method of claim 8, wherein the first pipeline status does not allow context switching where a context switch disable bit is set, the context switch disable bit being dynamically configurable during thread operation.

28. (Currently amended) ~~The computer processing method of claim 1~~, A computer processing method comprising the acts of:

using a shared pipeline instruction datapath and a shared pipeline processing unit to execute a first pipeline under a first context;

receiving a request to execute under a second context and, in response to the request, enabling a second pipeline to execute under the second context, wherein the enabling occurs during execution of the first pipeline, the enabling including fetching an address vector associated with the second context;

detecting a halt in execution of the first pipeline; and

using the shared datapath and shared processing unit to execute the second pipeline after detecting the halt;

wherein a second pipeline status allows context switching with any pipeline that does not have the second pipeline status, the second pipeline status being dynamically configurable during thread operation.